L'évolutionnisme pour l'aménagement du territoire

Sommaire

Introduction

Pour faire face à la complexité (cf. annexe B) de certains problèmes, comme ceux des transports appartenant à la classe NP, des heuristiques[1] permettent d'obtenir des

[1] Une heuristique est un algorithme fournissant une solution à un problème. Cette solution doit nécessairement être réalisable, mais pas forcément la meilleure.

solutions, pas nécessairement optimales mais répondant néanmoins à un ensemble de conditions. Les algorithmes génétiques, qui nous intéressent plus particulièrement dans cette thèse, appartiennent aux *métaheuristiques*. Celles-ci sont des heuristiques, dont le processus général est invariant quelle que soit l'application *in fine*.

Ce chapitre est donc principalement orienté sur l'algorithmique génétique. Toutefois, nous présentons d'abord plusieurs techniques dites « inspirées de la nature », dont les réseaux de neurones, les colonies de fourmis, les essaims particulaires et les algorithmes évolutionnistes. Il ne s'agit pas là de faire un état de l'art complet de ces méthodes mais de donner une vue suffisamment large sur les techniques inspirées de la nature pouvant contribuer à optimiser des problèmes de géographie (cf. section 5.2.2).

La deuxième partie de ce chapitre est intégralement dévolue aux algorithmes génétiques. Une description exhaustive de leur mécanisme ainsi que des exemples d'application détaillés sont fournis. Nous terminons en présentant l'algorithme NSGA-II, qui sert à développer une méthode d'optimisation en multiconvergence (cf. section 6.2).

5.1 Le concours de la nature à l'intelligence artificielle

C'est à partir des années 1950 que l'on voit croître un intérêt pour l'étude de la nature et de ses capacités d'adaptation et de reconfiguration vis-à-vis de quelque problème que ce soit. Cette première section est consacrée à une rapide revue de plusieurs techniques existantes (Cornuéjols et Miclet, 2002) dont les algorithmes évolutionnistes, outils de notre étude sur les TAD.

Sont ainsi décrits dans un premier temps les réseaux de neurones, les colonies de fourmis, les essaims particulaires, trois techniques majeures de l'intelligence artificielle inspirées de la nature.

5.1.1 Les réseaux de neurones

Également appelés réseaux neuromimétiques ou encore réseaux connexionnistes, les réseaux de neurones s'inspirent du fonctionnement du cerveau et de ses connexions synaptiques pour produire des modèles capables de tirer des conclusions à partir de données en entrée.

Unités de traitement très simples, les cellules nerveuses constituant le cerveau sont par-contre extrêmement nombreuses (environ cent milliards de neurones pour un cerveau humain), et bénéficient d'une interconnexion très dense (aux alentours de dix

mille connexions par neurone). Le mécanisme d'apprentissage et le comportement du réseau, issus de cette schématisation, sont déterminés par l'architecture et le nombre de neurones, et leurs connexions. Une connexion entre deux neurones se distingue par son *poids*, i.e. un arc indiquant le degré d'influence d'un neurone vers un second. La modification des poids de ces connexions revient à modifier la connaissance générale d'un problème, autrement dit à tirer de nouvelles conclusions avec des données identiques en entrée.

Initialement, un jeu d'exemples permet de mettre en mémoire ces nouvelles informations. Ainsi, l'apprentissage d'un réseau de neurones se fait par propagation d'informations échangées entre des unités élémentaires aux possiblités restreintes, mais aux interconnexions très développées, permettant de résoudre des problèmes complexes.

Le neurone formel

Les données d'entrée sont des vecteurs $x = (x_1, ..., x_n)$ de $\mathbb{R}^n$. Nous nous intéressons aux réseaux *multicouches*.

Description : Un neurone formel n'est capable que de quelques opérations élémentaires. Dans un réseau, les neurones sont classés selon trois types :

- les neurones d'*entrée* prenant en charge le vecteur initial x^0, et constituant la couche d'entrée ;
- les neurones de *sortie* fournissant des hypothèses d'apprentissage. C'est la couche de sortie ;
- les neurones *cachés*, n'étant ni des neurones d'entrée, ni des neurones de sortie.

État : Chaque neurone a un état de fonctionnement permettant ainsi de décrire le réseau à un moment précis. Cet état est calculé par une règle de propagation. Un neurone i d'entrée a pour état $\sigma_i = x_i$.

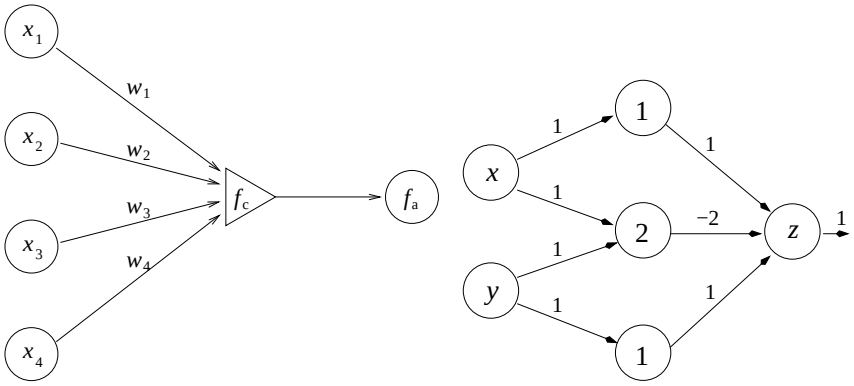
Fonctionnement : Un neurone dispose d'une fonction de sortie f calculant la valeur de sortie y_i en fonction de l'état σ_i du neurone i : $y_i = f(\sigma_i)$.

La figure 5.1(a) donne une vue simplifiée d'un réseau de neurones à quatre entrées. La figure 5.1(b) offre quant à elle le schéma de fonctionnement du *perceptron* reproduisant le fonctionnement de l'opérateur logique booléen XOR (« *ou exclusif* », $\oplus$, table de calcul 5.1).

Notons par ailleurs qu'il existe un seuil d'activation pour chaque neurone, permettant de propager une valeur si et seulement si ce seuil est atteint. Ainsi dans l'exemple du perceptron, la valeur -2 est propagée seulement lorsque le seuil fixé à 2 est atteint, i.e. lorsque x et y valent chacun 1 et que leurs valeurs sont cumulées.

x	y	$\oplus$
0	0	0
0	1	1
1	0	1
1	1	0

TAB. *5.1:* *Opérateur logique XOR* $\oplus$



(a) Réseau à quatre entrées (b) Le perceptron

FIG. *5.1:* *Exemples de réseaux de neurones*

Apprentissage

Les réseaux de neurones sont multicouches, c'est-à-dire que chacune des couches i, à l'exception de la dernière, constitue l'entrée de la couche $i + 1$.

Le mécanisme d'apprentissage s'effectue grâce à des jeux de tests fournis comme bases de connaissances, à partir desquelles le réseau tend à généraliser des hypothèses. Une fois ces hypothèses clairement posées, le réseau est *a priori* en mesure de conclure sur tout problème posé. La connaissance du réseau de neurones est distribuée dans les poids des connexions neuronales. Ces poids sont fixés par apprentissage (exemple par rétropropagation de gradient pour le perceptron).

Supervision : On peut forcer le réseau à généraliser et à tendre vers un vecteur final précis. On parle dans ce cas d'« *apprentissage supervisé* ». *A contrario*, si le réseau tire lui-

même ses conclusions, i.e. tend vers un vecteur final quelconque, l'apprentissage est dit « *non-supervisé* ».

Surapprentissage : Le réseau tire des conclusions à partir d'exemples fournis initialement. Supposons que les données prises en base d'apprentissage soient approximatives voire confuses, le réseau de neurones se trouve dans ce cas incapable de statuer une décision pour un problème dont les paramètres seraient tout aussi confus que les jeux de test.

Bruit à l'apprentissage : De même, il arrive qu'en donnant des connaissances de base, le réseau en tire de mauvaises hypothèses, non pas en raison d'un caractère erronné de ces informations, mais à cause d'informations parasites qui n'étaient pas anticipées. On parle dans ce cas de bruit.

Limites

Les réseaux connexionistes se prêtent peu voire pas du tout à la résolution de problèmes de tournées, notamment en raison du besoin de connaissances initiales à travers des exemples représentatifs et à l'effet « boîte noire » du à la méconnaissance sémantique des poids de la couche cachée des neurones.

5.1.2 Les colonies de fourmis

S'inspirant du comportement collectif des fourmis, Marc Dorigo propose dans les années 1990 des algorithmes à métaheuristiques (Dorigo, 1992) pour trouver les chemins optimaux dans un graphe. Bien que leur comportement et leurs capacités soient très limités, les fourmis sont capables de résoudre des problèmes complexes grâce à la collaboration des unes avec les autres. On retrouve ici l'idée que l'on avait déjà avec les réseaux de neurones où l'élément de base est très limité mais son interconnexion avec les autres éléments permet de résoudre simplement des problèmes. Ici, l'élément de base est la fourmi, et celle-ci n'est capable que de déposer des phéromones, qui se révèlent volatiles avec le temps.

Recherche du meilleur chemin

En observant le chemin emprunté par une colonie de fourmis pour se rendre à une source de nourriture, l'on se rend vite compte que celles-ci exploitent le chemin optimal. Leur comportement peut se résumer au schéma suivant :

1. Une fourmi explore son environnement aléatoirement, et sitôt qu'elle trouve une source de nourriture elle rentre au nid en marquant son chemin de retour de phéromones ;
2. Les fourmis à proximité des phéromones les ressentent et remontent la piste jusqu'à la source de nourriture. De retour au nid, elles déposent également des phéromones attractives renforçant la piste précédemment empruntée par leurs congénères ;
3. Si plusieurs chemins sont marqués par les phéromones attractives, le chemin le plus court sera également le chemin le plus marqué, car durant le même temps de parcours, plus de fourmis auront renforcé la piste ;
4. Au cours du temps, les phéromones se dispersent, et comme les chemins les plus marqués en phéromones (les plus courts) sont également les plus entretenus, les chemins les moins marqués (les plus longs) disparaissent progressivement.

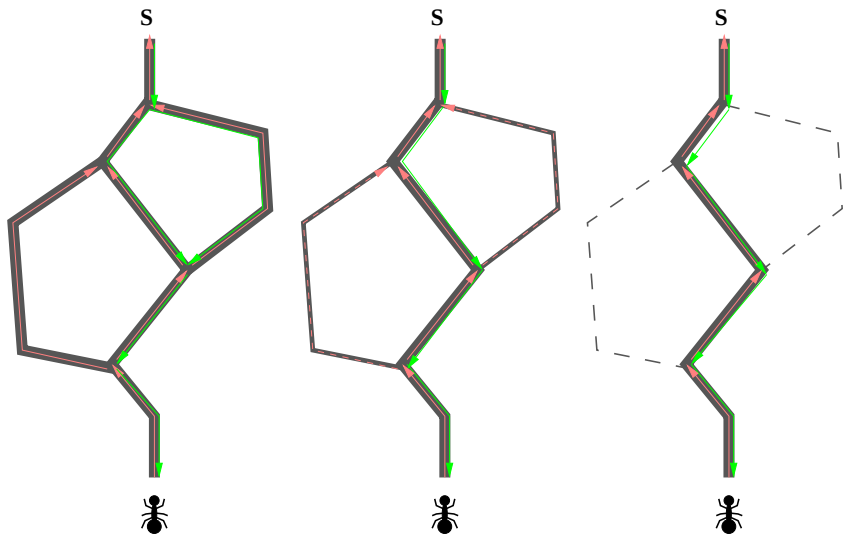
Avec un comportement somme toute assez simpliste, les fourmis ont été capables d'isoler rapidement le meilleur chemin pour assurer leurs besoins.

L'échange d'informations se fait *via* l'environnement et cet échange a une portée locale, en effet seules les fourmis à proximité des phéromones prennent connaissance de ces informations. Leurs diffusions se font par propagation successive suite aux dépôts successifs de phéromones.

Dans l'exemple de la figure 5.2, une colonie de fourmis explore tous les chemins possibles pour se rendre vers S (5.2(a)). Petit à petit, au fil des évaporations des phéromones sur les moins bons chemins et à la persistence des phéromones sur les meilleurs chemins (5.2(b)), les fourmis finissent par isoler un chemin optimal, celui qui est seul marqué par les phéromones (5.2(c)).

Limites

Bien que les colonies de fourmis soient adaptées à la recherche de chemins optimaux, ces algorithmes n'offrent pas de solutions multiples en une simulation. Pour offrir plusieurs solutions, l'on doit recourir à un ensemble de simulations.



(a) Étape initiale (b) Étape intermédiaire (c) Étape terminale

FIG. 5.2: *Une colonie de fourmis cherche le meilleur chemin pour accéder à une source de nourriture.*

5.1.3 Les essaims particulaires

Autres algorithmes à métaheuristiques reproduisant les comportements des insectes sociétaux ou des animaux vivant en groupe, comme les essaims d'abeilles ou les bancs de poissons, les essaims particulaires partent du principe que le groupe reproduira le comportement d'un invidividu qui semble avoir trouvé la meilleure solution à un instant précis (Kennedy et Eberhart, 1995). Imaginons un banc de poissons, dans ce groupe un individu détecte la présence d'un prédateur et emprunte la direction de déplacement opposée pour fuir le danger. Ce comportement est propagé aux autres individus qui adoptent quasiment aussitôt la même attitude.

D'autre part, ce même essaim peut adopter une attitude exploratoire, i.e. chaque particule dispose d'une marge de déplacement aléatoire pour explorer son environnement. Mais le comportement général demeure influencé par l'ensemble des particules. Les individus collaborent donc entre eux.

Pour résoudre un problème, les particules sont positionnées aléatoirement ou non dans l'espace de recherche des solutions et leur sont affectées des vitesses de déplacement (*vélocité*). Chaque particule connaît sa meilleure position et la position optimale de l'essaim. Ensuite, les particules communiquent à leurs proches leurs bonnes positions pour que chaque particule se repositionne et adapte sa vélocité compte tenu des meilleures indications.

Fonctionnement

Description Nous faisons évoluer un essaim de n particules dans un espace de résolution $\mathbb{R}^m$. Chaque particule i ($i \in \mathbb{N}, i \leq n$) a une position x_i associée, $x_i \in \mathbb{R}^m$, et une vélocité $v_i \in \mathbb{R}^m$. Notons x_i^+ la meilleure position connue de chaque particule et x_g^+ la meilleure position connue de l'ensemble du groupe.

L'algorithme canonique d'optimisation par essaim particulaire Nous utilisons une fonction objectif $f : \mathbb{R}^m \rightarrow \mathbb{R}$. Une meilleure solution minimise le score, i.e. x_i est meilleur que x_j si $f(x_i) < f(x_j)$. ω est une constante d'inertie. Les constantes k_1, k_2 indiquent dans quelle mesure la particule se dirige vers de bonnes positions. Elles représentent les composantes cognitive et sociale et interviennent sur le mouvement (respectivement) de la particule et du groupe. r_1, r_2 sont des vecteurs aléatoires dont les valeurs sont comprises entre 0 et 1.

1. Initialisation de x_i et v_i
2. Initialisation de $x_i^+ : x_i^+ \leftarrow x_i$, et de $x_g^+ : x_g^+ \leftarrow \min_i^n f(x_i)$
3. **Tant que** le critère d'arrêt n'est pas atteint **faire**
   Pour chaque particule **faire**
   (a) $x_i \leftarrow x_i + v_i$
   (b) $v_i \leftarrow \omega v_i + k_1 r_1 \times (x_i^+ - x_i) + k_2 r_2 \times (x_g^+ - x_i)$
   (c) **si** $f(x_i) < f(x_i^+)$ **alors** $x_i^+ \leftarrow x_i$
   (d) **si** $f(x_i) < f(x_g^+)$ **alors** $x_g^+ \leftarrow x_i$
   Fait
   Fait

Limites

Les flots de particules, dans leur forme de base sont peu adaptés aux problèmes à objectifs multiples comme le TAD, à cause notamment de leur recherche de solutions grégaires.

5.1.4 Les algorithmes évolutionnistes

Librement inspirés des théories évolutionnistes de Darwin, les algorithmes évolutionnistes reproduisent les processus d'adaptation des espèces en fonction de leur en-

vironnement, c'est-à-dire qu'au sein d'une espèce, certains de ses individus présentent des éléments physiques et/ou comportementaux plus appropriés à leur milieu. Ainsi favorisés par ces particularités et plus aptes à survivre à leur environnement, ces individus seront également favorisés dans le processus de repoduction et transmettront tout ou partie de leur patrimoine génétique à leurs enfants. C'est la sélection naturelle.

Ainsi sélectionnés au fil des générations, les individus seront plus adaptés à leur environnement car bénéficiant d'un patrimoine génétique plus approprié, hérité de seulement quelques aïeux plusieurs générations auparavant.

Le mécanisme des algorithmes évolutionnistes consiste donc à faire évoluer une population de solutions (les individus) toutes plus ou moins bonnes ou mauvaises et à diriger l'évolution de la population en fonction d'un objectif à atteindre, pour que les solutions soient plus adaptées au problème.

Les algorithmes évolutionnistes regroupent plusieurs types d'algorithmes dont les algorithmes génétiques qui nous intéressent plus particulièrement.

5.2 L'algorithmique génétique : fondements et principes

5.2.1 Origine de l'algorithmique génétique

Les premiers pas en algorithmique génétique furent initiés durant les années 1970 par John Holland et son équipe de l'Université du Michigan. En 1975, Holland publie « *Adaptation In Natural And Artificial Systems* » consacré à l'algorithmique génétique. À partir de ce moment, d'autres équipes vont lui emboîter le pas et contribuer activement à cette nouvelle thématique de recherche très prometteuse. Parmi ces grands laboratoires, l'on peut citer IlliGAL (*Illinois Genetic Algorithms Laboratory*) où travaille David Goldberg, EClab (*Evolutionary Computing laboratory*, Université George Mason) avec Kenneth A. DeJong, ou encore KanGAL (*Kanpur Genetic Algorithms Laboratory*), dirigé par Kalyanmoy Deb.

Ces équipes ont introduit quelques unes des notions majeures des algorithmes génétiques (AG), que l'on retrouve désormais de manière récurrente dans chacune de leurs utilisations[2], que ce soit en physique avec des systèmes multi-agents (Hippolyte *et al.*, 2007), en génomique pour l'apprentissage de règles (Jourdan *et al.*, 2002) ou encore en bioinformatique (Jourdan *et al.*, 2007)...

[2] À l'exception, par exemple, des algorithmes mémétiques.

5.2.2 L'algorithmique génétique en géographie

Les capacités d'optimisation des AG ainsi que leur capacité à offrir plusieurs solutions à la fois ont particulièrement intéressé les géographes et les aménageurs, anglo-saxons pour la plupart.

Les AG sont donc utilisés dans des problèmes d'aménagement de parcelles agricoles et de développement durable avec pour forte contrainte de limiter les émissions de CO_2 (Datta *et al.*, 2006).

Les problèmes de localisation (Xiao, 2006) ou encore l'automatisation de la détection de regroupement (Conley *et al.*, 2005) ont incité les géographes à se pencher sur les AG. D'une manière générale, ce sont les problèmes d'optimisation non linéaires comme ceux étudiés par Gallagher et Sambridge (1994) qui ont suscité l'intérêt pour ces algorithmes.

En France, nous pouvons citer les travaux de (Bolot *et al.*, 1999), dans la recherche de découpage territorial optimal, qui ont donné lieu à ce moment là à un concours pluridisciplinaire.

En outre, une force des AG réside dans leur robustesse, ce qu'Openshaw (1997) n'a pas manqué de souligner.

5.2.3 Analogie avec la biologie

La terminologie employée pour les AG fait clairement référence à celle de la biologie (Goldberg, 1989; Whitley, 1994; DeJong, 2006).

Population et individus

Un AG rassemble un nombre $n \geq 2$ de solutions au problème posé. Ces solutions sont des *individus*, encore appelés *chromosomes*, et constituent une *population*. Plus la population est importante, plus le patrimoine génétique est *a priori* important (cela dépend aussi de l'initialisation), cependant l'évolution sera plus lente[3].

[3] En temps de calcul sur ordinateur.

Gènes, locus et allèles

Un chromosome regroupe un ensemble d'attributs ordonnés appelés *gènes*. Un gène prend pour valeur l'une parmi celles possibles : ces valeurs sont les *allèles*. L'emplacement d'un gène s'appelle *locus*. Dans le cas d'un encodage binaire les allèles possibles sont $V = \{0, 1\}$.

$$\mathcal{P} = \begin{cases} A : & a \mid b \mid \ldots \mid i \mid j \\ B : & a \mid b \mid \ldots \mid i \mid j \\ & \ldots \\ N : & a \mid b \mid \ldots \mid i \mid j \end{cases}$$

FIG. 5.3: *Une population $\mathcal{P} = \{A, B, ..., N\}$ constituée de $|\mathcal{P}| = n$ individus ; les allèles possibles sont $V = \{a, b, ..., i, j\}$*

5.2.4 Mécanisme

La notion évolutionniste ou génétique d'un algorithme caractérise en fait l'imitation du processus naturel d'évolution d'une population. Un AG en reprend ses lignes directrices pour adapter un ensemble de solutions de manière optimale à l'environnement. Un AG est donc un processus itératif de recherche exploratoire dont les étapes sont les suivantes :

1. **Initialisation** de la population $\mathcal{P}_0$ (population de génération 0) ;
2. **Évaluation** des chromosomes de la population $\mathcal{P}_i$;
3. **Sélection** des meilleurs chromosomes de $\mathcal{P}_i$ qui vont constituer les « procréateurs » de la génération $\mathcal{P}_{i+1}$;
4. **Croisement** deux-à-deux des chromosomes sélectionnés pour générer de nouveaux individus, les « descendants » ;
5. **Mutation** opérée aléatoirement sur un chromosome également choisi au hasard pour apporter de la diversité dans le patrimoine génétique de la population ;
6. **Remplacement** et génération de la nouvelle population $\mathcal{P}_{i+1}$;
7. **Répéter** le processus à partir de 2 tant que le critère d'arrêt n'est pas satisfait (i.e. génération n terminale non atteinte, solution optimale non trouvée...).

Que représente un individu ?

Nous sommes invités à nous poser cette question pour encoder les paramètres du problème qui sont portés par un chromosome. Car derrière l'apparente simplicité d'un AG, l'on se rend vite compte que la réelle difficulté réside justement dans le choix de la bonne représentation.

Il n'y a pas de représentation absolue. L'encodage se fait au cas par cas selon les types de paramètres à prendre en compte. Ainsi l'encodage d'un problème peut aller de la simple chaîne binaire, à un arbre en passant par les listes chainées[4]... Cependant, à l'instar de la nature l'on préfère généralement utiliser des chromosomes simples sous forme de chaînes, i.e. en termes d'implémentation informatique il s'agit de tableaux unidimensionnels.

Considérons l'exemple suivant. Avec une feuille de papier aux dimensions $L_f \times l_f$ (respectivement longueur et largeur de la feuille f), nous souhaitons la plier pour obtenir la plus grande boîte possible. Le problème consiste donc à trouver les meilleures dimensions l_V, h_V, p_V (respectivement longueur, hauteur, profondeur du volume V) pour obtenir le plus grand volume $V = l_V \times h_V \times p_V$. Ne considérant que des dimensions entières, l'encodage peut se résumer à un tableau de trois valeurs (cf. tab. 5.2), $[l_V(i), h_V(i), p_V(i)]$. La figure 5.4(a) représente la feuille à plier et sur les figures 5.4(b,c,d) sont représentés les patrons de pliage associés aux chromosomes du tableau 5.2.

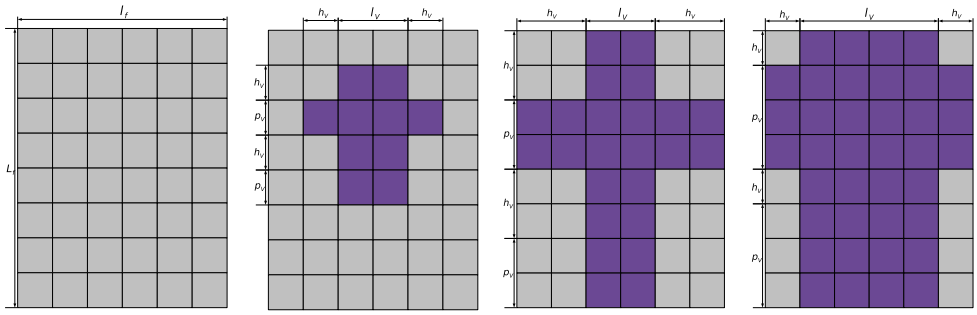
i	$l_V(i)$	$h_V(i)$	$p_V(i)$	$V(i)$	$fitness(i)$
1	2	1	1	2	2
2	2	2	2	8	8
3	4	1	3	12	12
4	2	2	3	impossible	0
5	4	1	4	impossible	0

TAB. 5.2: *Encodage du problème et plusieurs chromosomes*

Caractérisation d'une bonne solution

À chaque génération, les chromosomes sont évalués, et un score leur est attribué. Celui-ci est calculé à l'aide d'une fonction d'évaluation (*fitness function*) attribuant un score à un chromosome en fonction de la qualité de la solution apportée au problème sous contraintes des conditions formulées. Dans le cas de notre feuille de carton à plier,

[4]Structure informatique de données où chaque élément est lié au suivant.



(a) Feuille originale (b) Volume 2 (c) Volume 8 (d) Volume 12

FIG. 5.4: *Les patrons de pliage correspondant aux chromosomes du tableau 5.2 : trois solutions possibles dont une optimale (d) avec un scrore de 12.*

celle-ci a pour dimensions $L_f = 8$ et $l_f = 6$. Autrement dit, la largeur l_V ne doit pas dépasser la largeur de la feuille moins la hauteur du volume à créer, h_V (eq. 5.1). De plus, la profondeur p_V ne peut excéder la moitié de la longueur de la feuille L_f moins la hauteur h_V (eq. 5.2). Donc, une bonne solution consiste à maximiser V sous contraintes 5.1,5.2.

$$l_V \leq l_f - 2 \times h_V \tag{5.1}$$

$$p_V \leq \frac{L_f}{2} - h_V \tag{5.2}$$

Sélection

Soit p_j la probabilité qu'un chromosome j soit sélectionné. Cette probabilité peut être déterminée de différentes manières, soit elle dépend simplement du nombre de chromosomes (aléas), soit elle est relative à la qualité de la solution (roulette).

Aléa. Les chromosomes parents peuvent être choisis de manière totalement aléatoire parmi la population $\mathcal{P}$, $p_j = \frac{1}{|\mathcal{P}|}$.

La roulette. La méthode avec « roulette des scores » (*the score wheel method*) choisit aléatoirement un chromosome parmi la population, cependant celui-ci a une probabilité d'être choisi relative à son score. Plus celui-ci est élevé, plus sa chance d'apparition lors de la sélection est importante.

Considérons l'exemple suivant. Nous disposons d'une population de dix chromosomes $\mathcal{P} = \{C_0, ..., C_9\}$ aux scores positifs S_k (tab. 5.3 et fig. 5.5), et nous réalisons la roulette des choix correspondante selon la méthode suivante :

1. Sommation des scores : $\Sigma_{k=0}^{9} S_k$;
2. Déduction des probabilités d'apparition $p_j = \dfrac{S_j}{\Sigma_{k=0}^{9} S_k}$

Ainsi le chromosome C_4 de la population $\mathcal{P}_i$ noté $S_4 = 3$ a 15% de chances d'être sélectionné lors d'un croisement pour générer la population $\mathcal{P}_{i+1}$, tandis que le chromosome C_6 au score nul n'est pas retenu pour un prochain croisement.

Bien qu'il soit le meilleur chromosome et même avec 20% de chances d'être retenu pour un croisement, le chromosome C_0 pourrait potentiellement ne pas figurer dans un croisement ultérieur, alors pour éviter de perdre un aussi bon patrimoine génétique l'on pourrait avoir recours à une copie directe de ce chromosome dans la génération suivante. Cette optimisation figure généralement dans les algorithmes élitistes qui visent à promouvoir les meilleurs éléments au détriment des moins performants (cf. 5.2.6).

C	0	1	2	3	4	5	6	7	8	9	total
S	4	1	2	1	3	1	0	4	1	3	20
p	0.2	0.05	0.1	0.05	0.15	0.05	0.0	0.2	0.05	0.15	1.0

TAB. 5.3: *Probabilités d'apparition des chromosomes en fonction de leurs scores*

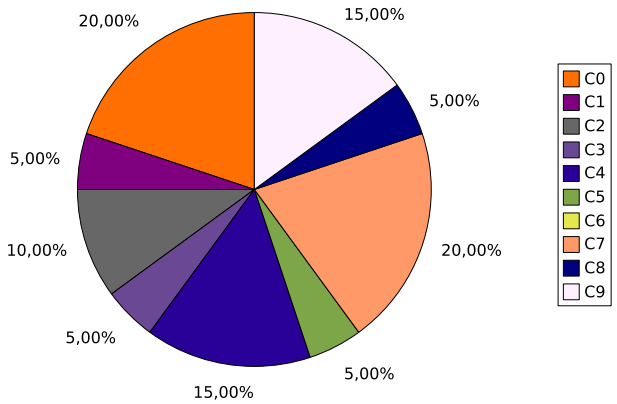


FIG. 5.5: *Roulette des scores possible pour sélectionner les chromosomes à croiser*

Croisement

Le croisement est une application de : $C \times C \rightarrow C \times C$ permettant de brasser la population. Il s'effectue sur des paires de chromosomes choisis durant la phase antérieure de sélection et consiste à reproduire les chromosomes des parents et à échanger des séquences de gènes des chromosomes, produisant ainsi de nouveaux individus au patrimoine génétique brassé. Nous nous intéressons plus particulièrement aux croisements suivants :

- croisement à un point : un point p est choisi aléatoirement (ou non, on peut échanger la moitié du chromosome comme dans le *Simple Genetic Algorithm* (Goldberg, 1989) sur des chromosomes de taille l, $p < l$, et les séquences (pl) sont échangées en produisant les chromosomes enfants de taille l (cf. fig 5.6).
- croisement à deux points : deux points p, q sont choisis aléatoirement sur des chromosomes de taille l, $p < q < l$, et les séquences (pq) sont échangées pour créer les enfants de taille l (cf. fig 5.7).

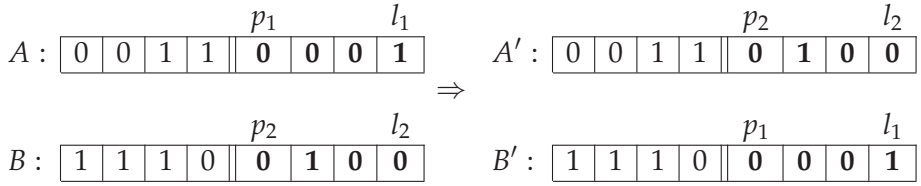


FIG. 5.6: *Croisement à un point de deux chromosomes binaires*

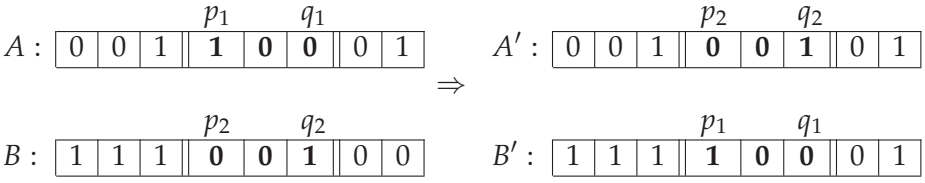


FIG. 5.7: *Croisement à deux points de deux chromosomes binaires*

Mutation

La mutation est une application de : $C \rightarrow C$ et a pour but d'introduire de la diversité dans une population (Holland, 1975). En effet, le croisement risque de faire converger la population vers un ensemble restreint de solutions. On parle dans ce cas de convergence vers un puits local de solutions. Une fois l'ensemble des chromosomes situé dans

cet espace de solutions, l'apparition de nouvelles solutions au patrimoine génétique diversifié devient plus difficile. Pour pallier ce problème, l'on introduit de la diversité par le biais de la mutation qui consiste à modifier un chromosome sous une probabilité de mutation p_{mut}. Le chromosome muté voit un de ses gènes choisi au hasard et modifié.

0	1	1	0	0	**0**	1	1

(a) Chromosome initial

0	1	1	0	0	**1**	1	1

(b) Chromosome muté

FIG. 5.8: *Exemple de mutation sur un chromosome binaire*

L'exemple de la figure 5.8 illustre un mécanisme très simple, mais la mutation peut servir davantage qu'à diversifier le patrimoine génétique du chromosome ; elle peut servir notamment à contribuer à la résolution d'un problème, par exemple en forçant la convergence (Chevrier *et al.*, 2006b) tout en maintenant la diversité !

Génération de la nouvelle population

Une fois la mutation effectuée, nous sommes presqu'au terme de l'itération i de l'algorithme génétique canonique, il reste maintenant à générer la population $\mathcal{P}_{i+1}$ à partir de la population $\mathcal{P}_i$ augmentée de sa descendance $\mathcal{E}$, ses enfants.

Diverses politiques existent, simple ou élitiste. Dans le cas d'un renouvellement de population simple, seuls les enfants générés constituent la nouvelle population (cf. fig. 5.9). En revanche, dans le cas d'un renouvellement élitiste, les chromosomes enfants sont ajoutés à la population temporaire elle-même déjà constituée des parents. Tous ces invidus sont évalués et classés par score. Seuls les n meilleurs chromosomes sont conservés (cf. fig. 5.10), pour revenir à la taille de population initiale (n individus).

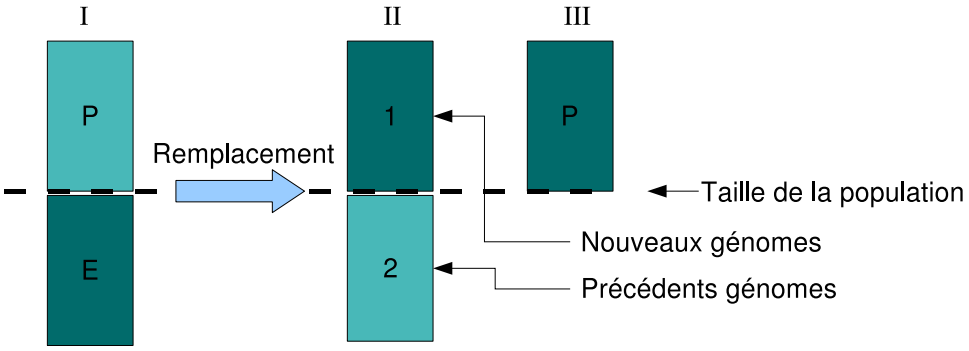


FIG. 5.9: *Vue simplifiée d'une politique de diversité dans un algorithme génétique*

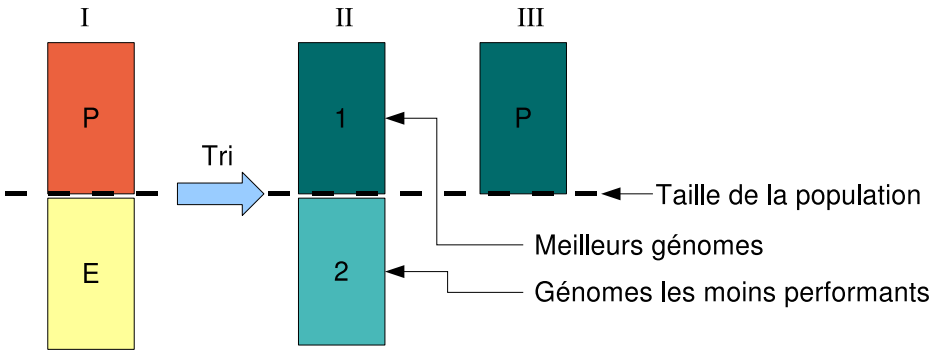


FIG. 5.10: *Vue simplifiée d'une politique élitiste dans un algorithme génétique*

5.2.5 Un algorithme génétique mono-objectif

Examinons maintenant plus en détails un exemple d'AG mono-objectif, qui nous rapproche un peu plus encore d'un problème de découpage territorial. Le problème des huit reines est un véritable cas d'école en cours d'algorithmique. Sur un échiquier de 8×8 cases, l'on souhaite placer 8 reines de telle sorte qu'aucune d'elles ne soit en prise avec une autre. Pour rappel, la reine, au jeu d'échecs, se déplace horizontalement, verticalement et en diagonale, sur plusieurs cases. Nous allons voir dans cette sous-section une manière de résoudre ce problème.

Définition du problème

La figure 5.11 donne une solution possible au problème : chaque reine est seule sur les couloirs formés horizontalement, verticalement et en diagonale, i.e. il n'y a nécessairement qu'une seule reine par ligne et par colonne, et au plus une seule reine par diagonale. Le problème revient donc à ajuster la bonne ligne à la bonne colonne.

Initialisation : une seule reine par colonne. Pour chaque colonne restante, on en choisit une aléatoirement (position x), et pour cette colonne, nous choisissons une position verticale y aléatoirement parmi les lignes restantes. Une fois le couple (x, y) positionnant une reine choisi, nous retirons la ligne et la colonne désignées de l'ensemble des lignes et colonnes restantes.

Objectif et évaluation De par l'initialisation à laquelle nous avons procédé (une seule reine par colonne), nous savons qu'il suffit de tester l'unicité d'une reine sur sa ligne et sur sa diagonale pour indiquer la validité de cette position. Il faut donc 16 tests pour évaluer un chromosome : 8 pour les lignes + 8 pour les diagonales. La politique d'évaluation échelonne les scores de 0, pour une solution nulle, à 1 pour une solution correcte remplissant toutes les conditions. À chacune des 16 conditions remplies, nous augmentons le *fit* Φ de $\frac{1}{16}$. Initialement, tous les chromosomes i disposent donc d'un *fit* $\Phi_i \geq \frac{1}{16} \times 8$, car il n'y a qu'une reine par ligne.

Représentation de la solution

Différentes manières de représenter une solution existent. L'on pourrait utiliser un simple chromosome unidimensionnel à l'instar de la feuille à plier, en associant un

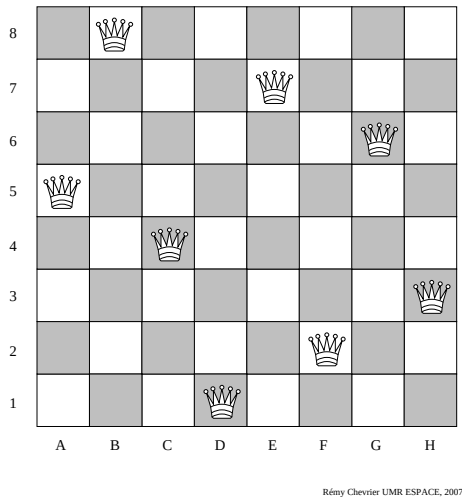


FIG. *5.11:* *Une configuration possible au problème des huit reines*

numéro de ligne pour chacune des cellules du tableau qui correspondent aux indices des colonnes (tab. 5.4). Nous pouvons également utiliser une représentation matricielle (assez proche finalement d'une carte rasterisée) binaire carrée (8×8), représentant directement l'échiquier, où 1 indique la présence d'une reine et 0 une case vide (tab. 5.5).

colonnes (A↔H)	1	2	3	4	5	6	7	8	
lignes (1↔8)	5	8	4	1	7	2	6	3	← chromosome

TAB. *5.4:* *Chromosome décimal unidimensionnel représentant la configuration de la figure 5.11*

0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0

TAB. *5.5:* *Chromosome binaire carré représentant la configuration de la figure 5.11*

Cependant, les informations utiles (les 1 représentant les reines) ne représentent que $\frac{1}{8}$ de la matrice. En définitive, pour ce problème, nous manipulons beaucoup plus d'informations inutiles. L'utilisation d'une matrice se révèle alors très coûteuse en espace mémoire mais aussi plus délicate à manipuler lors des croisements. Le vecteur, quant à lui, se révèle bien plus économique et demeure tout aussi expressif. Il est de surcroît plus simple à manipuler lors des croisements comme nous allons le voir.

Croisement et mutation

Croisement Le brassage de population modifie les positions des reines sur les lignes, aussi échange-t-on les colonnes de deux chromosomes C_A et C_B. Cet échange peut être effectué à l'aide d'un croisement à un point ou à deux points, bien adapté au besoin de cet algorithme. Nous utilisons un croisement à deux points qui consiste à échanger les séquences de deux chromosomes choisis selon une méthode basée sur la roulette des scores.

Le croisement occasionné génère deux nouveaux individus, les chromosomes $C_{A'}$ et $C_{B'}$.

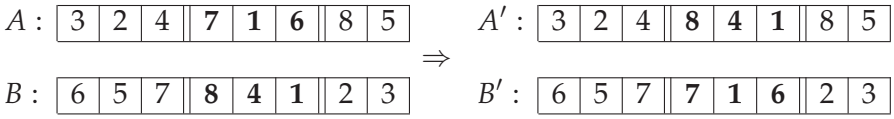


FIG. 5.12: *Croisement à deux points de deux chromosomes*

Mutation Le mécanisme de mutation (fig. 5.13) adopté est somme toute assez simple et consiste à modifier une valeur de gène aléatoirement sur le chomosome C_A. Si elle n'est pas dirigée, la mutation ne fait qu'apporter de la diversité dans un patrimoine génétique sélectionné au fur et à mesure des croisements, mais ne fait pas forcément émerger de bonnes solutions.

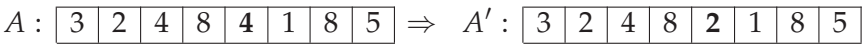


FIG. 5.13: *Mutation sur un gène*

5.2.6 Diversité contre élitisme

Un point majeur (sans oser dire crucial !) concerne le choix de la politique de sélection des chromosomes. Généralement en AG, soit l'on opte pour maintenir de la diversité dans la population, soit l'on préfère une stratégie élitiste de sélection visant à ne conserver que les meilleures solutions. Cependant, on a souvent à faire face au dilemne de la perte de patrimoine génétique au travers de mauvaises solutions qui, une fois croisées avec d'autres mauvaises solutions, pourraient potentiellement fournir des solutions plus avantageuses que les meilleures jusqu'alors.

En soi, il n'y a pas de stratégie meilleure qu'une autre. La question revient à savoir quelle est la politique de sélection la mieux adaptée au problème à résoudre.

Dans le détail, une politique visant à maintenir de la diversité ne sacrifie pas nécessairement les plus mauvaises solutions au détriment des meilleures. Le simple fait de les conserver nous assure un plus grand réservoir de gènes et donc, de solutions possibles. Cependant la convergence vers des solutions optimales ou sub-optimales risque d'être plus lente.

A contrario une politique élitiste force la conservation des meilleurs éléments de la population initiale et de la population générée pour constituer la nouvelle population. Dans ce cas la convergence vers des solutions optimales ou sub-optimales sera nécessairement plus rapide. Cependant, le risque majeur de cette stratégie est de faire converger les solutions vers un puits local de solutions sub-optimales au risque de ne plus en sortir en raison du manque du patrimoine génétique des solutions éconduites du processus de sélection.

5.3 Gestion d'objectifs multiples

5.3.1 Des objectifs souvent contradictoires

Lorsque l'on souhaite optimiser plusieurs objectifs φ_i, ceux-ci sont bien souvent en opposition, c'est-à-dire que favoriser l'un revient à pénaliser l'autre. Considérons l'exemple suivant. On souhaite former une parcelle rectangulaire de 30 ha et l'on cherche les dimensions maximales l et L. Bien évidemment il n'y a pas une seule solution optimale, mais une infinité de solutions, toutes réparties sur la courbe de la figure 5.14.

Représenter l'ensemble des solutions disposées sur une courbe ($n = 2$ dimensions), sur une surface ($n = 3$), ou sur un front ($n > 3$) revient à utiliser une approche dite avec front de Pareto (Zitzler et Thiele, 1999). Les solutions représentées sous la forme $(x_1, ..., x_n)$ appartenant au front représenté dans l'espace des solutions à n dimensions sont dites Pareto-optimales. Les solutions qui ne sont pas situées sur le front, et néanmoins correctes, sont dites sub-optimales.

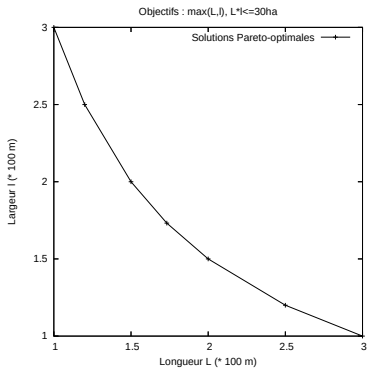


FIG. 5.14: *Solutions Pareto optimales au problème de constitution d'une parcelle de 30 ha, avec pour contraintes : $max(L, l)|L \times l \leq 30$*

5.3.2 Les algorithmes évolutionnistes multi-objectifs et la Pareto-dominance

Lorsque plusieurs objectifs sont évalués de manière concurrente, la fonction d'évaluation Φ associe un vecteur de scores pour chacun des m objectifs $(\varphi_1, ..., \varphi_m)$. Pour ordonner les chromosomes, une première solution consiste en une somme pondérée linéaire des différents scores attribués aux objectifs respectifs. Cependant, cette méthode risque d'occulter des solutions optimales du fait de l'incomparabilité de données différentes, car quel sens pourrions-nous donner à une somme de véhicules et de temps ?

Au cours des années 90, les problèmes multi-objectifs ont suscité le développement d'algorithmes génétiques traitant de manière concurrente ces objectifs. Une première généralisation des AG aux problèmes multi-objectifs est proposée par Fonseca et Fleming (1993) et c'est peu de temps après, que l'approche Pareto est introduite dans le processus de sélection (Horn *et al.*, 1994). L'approche Pareto permet en outre de conserver une grande variété de solutions en explorant plus largement les ensembles de solutions et en évitant les puits locaux, tout en introduisant la notion de *dominance* déjà proposée par Goldberg en 1989. On parle dans ce cas de *Pareto dominance* (Zitzler et Thiele, 1998a,b).

Comme nous l'avons mentionné préalablement, le tri de solutions issues d'un algorithme multi-objectifs pose le problème de discrimination des solutions les unes par rapport aux autres. En effet, dans notre AG à deux objectifs φ_1, φ_2 (à minimiser), avec lequel nous souhaitons ordonner deux chromosomes $C_a(\varphi_{1a}, \varphi_{2a})$, $C_b(\varphi_{1b}, \varphi_{2b})$, plusieurs cas de figures se présentent :

- si $\varphi_{1a} < \varphi_{1b}$ et $\varphi_{2a} < \varphi_{2b}$ alors C_a *domine* C_b : $C_a \prec C_b$;
- si $\varphi_{1a} > \varphi_{1b}$ et $\varphi_{2a} > \varphi_{2b}$ alors C_b *domine* C_a : $C_b \prec C_a$;
- si $\varphi_{1a} < \varphi_{1b}$ et $\varphi_{2a} > \varphi_{2b}$ alors on ne peut pas statuer quel est le meilleur chromosome.

Une première version d'algorithme avec approche Pareto et tri des individus non-dominés est donnée en 1995 : le NSGA (*Non dominated Sorting Genetic Algorithm*) de Srinivas et Deb (1995). Cependant, cet algorithme souffre d'une complexité atteignant $O(mN^3)$ avec N individus et m objectifs.

Par ailleurs, des résultats fournis dans (Rudolph, 1999; Zitzler *et al.*, 2000) indiquent que l'élitisme favorise la convergence vers les solutions optimales dans le cas des algorithmes multi-objectifs. Ainsi une variante d'élitisme est proposée dans *Pareto archived evolution strategy* (PAES) qui conserve les individus non-dominés dans une population annexe (Knowles et Corne, 1999). Zitzler et Thiele (1998b) fournissent également un algorithme élitiste avec approche Pareto : le *Strength Pareto Evolutionary Algorithm* (SPEA).

Ainsi cet ensemble de résultats a conduit à l'élaboration d'une seconde version de NSGA (NSGA-II) qui intègre un ensemble de modifications pour d'une part réduire la complexité et d'autre part intégrer l'élitisme dans la politique de sélection et de génération des populations. C'est l'objet de la sous-section 5.3.3.

5.3.3 Une méthode élitiste avec front de Pareto : NSGA-II

Le *Non dominated Sorting Genetic Algorithm II* (NSGA-II) proposé par Deb et son équipe est un AG utilisant l'approche Pareto (Deb, 2001), établissant les rapports de dominance entre les individus (Deb *et al.*, 2000) et offrant une méthode de tri particulièrement rapide des chromosomes. La complexité[5] totale atteint $O(mN^2)$ avec m objectifs et N chromosomes. La figure 5.15 donne une vue simplifiée de l'algorithme.

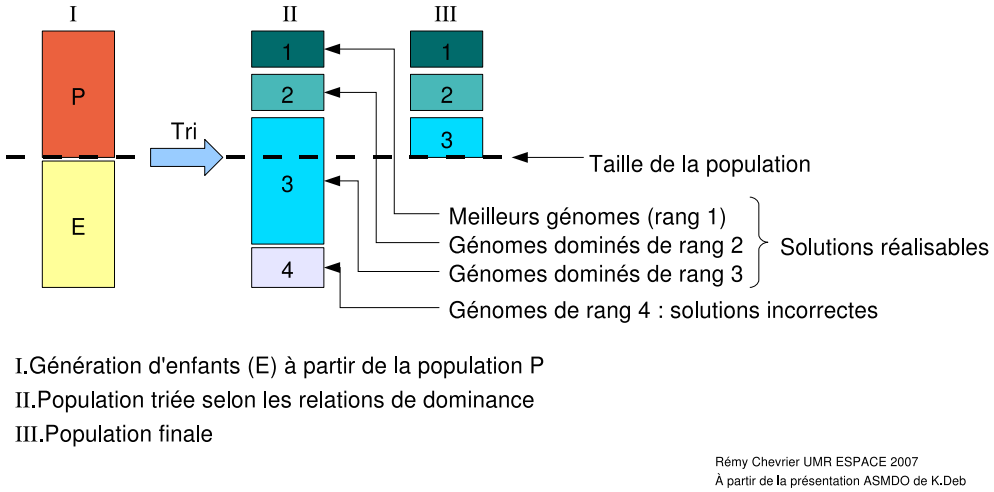


FIG. 5.15: *Vue simplifiée du fonctionnement de NSGA-II*

Tri des individus non dominés

Critiqué pour la complexité de sa première version ($O(mN^3)$) qui utilisait un algorithme de tri insuffisamment rapide, NSGA, dans sa seconde version, intègre une nouvelle méthode de tri plus rapide que nous détaillons ici.

Pour chaque solution, NSGA-II calcule n_i le nombre de solutions, qui dominent la solution i et S_i l'ensemble de solutions que domine i. Toutes les solutions ayant $n_i = 0$ sont ajoutées dans $\mathcal{F}_1$ qui correspond au front de solutions courant. Ensuite pour chaque solution du front courant, on examine chaque membre j de l'ensemble S_i en réduisant n_j de 1 pour le faire tomber à 0. De cette manière, si pour chaque membre j le compte tombe à 0, nous ajoutons j à une liste séparée $\mathcal{H}$. Lorsque tous les membres du front courant ont été passés en revue, les membres de la liste $\mathcal{F}_1$ appartiennent au premier front. Le processus est répété sur le nouveau front courant qui est l'ensemble $\mathcal{H}$.

[5] Au sens de la *calculabilité* du problème (cf. annexe B).

Chaque itération requiert $O(N)$ calculs et comme il y a au plus N fronts, la complexité de la boucle atteint $O(N^2)$. La complexité totale de l'algorithme atteint $O(mN^2) + O(N^2) = O(mN^2)$, avec m objectifs.

Algorithme de tri des individus non dominés rapide

1. **pour chaque** $p \in \mathcal{P}$ **faire**
   (a) **pour chaque** $q \in \mathcal{P}$ **faire**
   si $p \prec q$ **alors** # p domine q
   $S_p = S_p \cup q$ # ajout de q à l'ensemble des solutions dominées par p
   sinon si $q \prec p$ **alors** # q domine p
   $n_p = n_p + 1$ # on incrémente le compteur de solutions dominant p
   fin si
   fait
   (b) **si** $n_p = 0$ **alors** # pas de dominant à p
   $\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$ # ajout de p au front $\mathcal{F}_1$
   fin si
   fait
2. $i = 1$
3. **tant que** $\mathcal{F}_i \neq \oslash$ **faire** # construction des fronts de solutions (par rang)
   (a) $\mathcal{H} = \oslash$
   (b) **pour chaque** $p \in \mathcal{F}_i$ **faire**
   pour chaque $q \in S_p$ **faire**
   i. $n_q = n_q - 1$
   ii. **si** $n_q = 0$ **alors** $\mathcal{H} = \mathcal{H} \cup \{q\}$ **fin si** # q n'a plus de dominant donc on l'ajoute au front $\mathcal{H}$
   fait
   fait
   (c) $i = i + 1$ # l'ensemble $\mathcal{H}$ est terminé
   # et constitue le nouveau front de solutions
   (d) $\mathcal{F}_i = \mathcal{H}$ # $\mathcal{H}$ devient le front $\mathcal{F}_i$
   fait

Estimation de la densité de solutions

Estimer la densité des solutions revient à savoir si l'AG a fait converger les individus sur un ensemble restreint de solutions. Pour obtenir cette estimation, il faut calculer la distance moyenne des deux points situés de part et d'autre du point que l'on traite. Cette distance correspond à la longueur moyenne du côté du cuboïde encadrant le point examiné. Il correspond au rectangle pointillé de la figure 5.16.

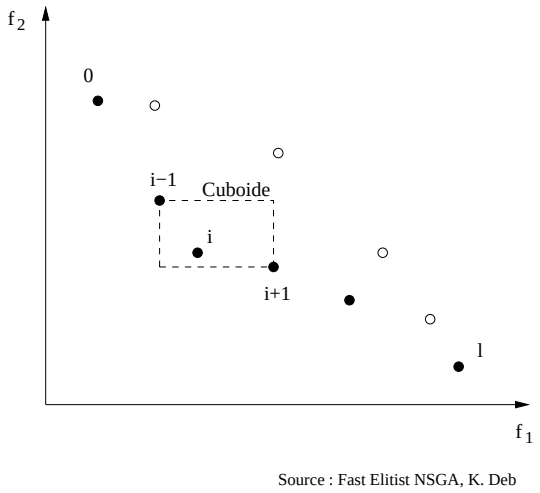


FIG. *5.16:* *Exemple de calcul de la distance d'un point à la population.*

L'algorithme suivant décrit l'affectation des distances moyennes de chaque individu de l'ensemble $\mathcal{I}$ ($\mathcal{I}[i+1].m$ représente la valeur de l'objectif m pour l'individu i dans la population $\mathcal{I}$) :

1. $l = |\mathcal{I}|$ # nombre de solutions dans $\mathcal{I}$
2. **pour chaque** i **faire** $\mathcal{I}[i]_{distance} = 0$ # initialisation de la distance
3. **pour chaque** objectif m **faire**
   (a) $\mathcal{I} = \text{tri}(\mathcal{I}, m)$ # tri selon l'objectif m :
   (b) $\mathcal{I}[1]_{distance} = \mathcal{I}[l]_{distance} = \infty$ # les points limites sont connus
   (c) **pour** i de 2 à $l-1$ **faire**
   $\mathcal{I}[i]_{distance} = \mathcal{I}[i]_{distance} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)$
   fait

fait

Ce calcul sert notamment à maximiser les distances entre les individus. Cela a pour effet d'éviter plusieurs occurrences du même individu ou du moins de chercher des solutions plus dissemblables. Le deuxième effet est d'étendre le front de Pareto et de

recouvrir plus de solutions différentes. Par ailleurs, cela permet d'éviter à l'ensemble des solutions de rester bloqué sur un puits local de solutions.

Conclusion

Ce chapitre donne lieu à un descriptif des méthodes inspirées de la nature et fournit un aperçu de leurs possibilités en matière d'optimisation. L'objectif est de montrer l'intérêt que présentent ces méthodes dans les problèmes d'optimisation géographiques, tels que ceux présentés dans la section 5.2.2.

Les algorithmes génétiques retiennent notre attention, notamment pour leur capacité à fournir plusieurs solutions en une optimisation. Car dans un contexte opérationnel, le décideur doit avoir le choix entre plusieurs alternatives équivalentes (au sens Pareto du terme).

C'est d'ailleurs pour cette même raison que nous avons fait évoluer l'AG, d'une optimisation mono-objectif (agrégat d'objectifs) vers une optimisation multi-objectifs avec une approche Pareto. Dans cette optique, plusieurs algorithmes existent et notre choix se porte sur NSGA-II, pour des raisons de rapidité d'une part et pour sa capacité à explorer tout le front de solutions d'autre part. Les algorithmes génétiques développés pour cette thèse sont présentés dans le chapitre suivant.